МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Стеки

Студент гр. 8304 _____ Ивченко А.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2019

Цель работы.

Ознакомиться с абстрактными типами данных; создать шаблонный класс стека на базе вектора(массива); определить методы, необходимые для работы над ним, включая деструктор и конструктор класса.

Задание(вариант 8-в).

В заданном текстовом файле F записано логическое выражение (ЛВ) в следующей форме:

<ЛВ>::= true | false | (!< ЛВ >) | (< ЛВ > V < ЛВ >) | (< ЛВ > ^ < ЛВ >)

где знаки !, V и ^ обозначают соответственно отрицание, конъюнкцию и дизъюнкцию. Вычислить (как Boolean) значение этого выражения.

Описание работы алгоритма.

Был создан шаблон класса *TEMPLATE <TYPENAME T> CLASS VSTACK* с приватными полями: память, выделенная под массив; длина массива; индекс верхнего элемента стека. Краткое описание публичных полей шаблона класса:

VSTACK(INT LENGTH) — конструктор класса

INT EMPTY() CONST — метод, проверяющий стек на пустоту

VOID PUSH(T ELEM) — функция добавляет элемент на вершину стека

VOID POP() - функция удаляет верхний элемент

T GET_TOP() - метод, возвращает верхний элемент типа данных t

VOID PRINT_STACK() - функция вывода стека на экран

Краткое описание функций.

void readLogicalExpr(vStack<bool> &stack, std::stringstream &ss) — функция считывает строку из потока данных, попутно проверяя строку на корректность. В случае считывания открывающей скобки функция вызывает саму себя до тех

пор, пока не будет обработано последнее ЛВ, заключенное в скобках. Таким образом, глубина рекурсии определяется количеством вложенных ЛВ. В стек сначала записывается первый операнд, затем второй(если он есть).В случае считывания закрывающей скобки функция возвращается на один уровень рекурсии выше, из стека вынимается два верхних элемента и добавляется результат операции между ними.

bool execOperation(char sign, bool op1, bool op2) — функция возвращает значение boolean между операндами op1 и op2.

bool readOperand(std::stringstream &ss, char cur) — функция определяет значение boolean операнда**.**

Тестирование.

```
(!false)= 1
(false V true)= 1
(false^(false^true))= 0
(!(false^false))= 1
((!true) V (!(falseVtrue)))= 0
(true ^ (!(falseV(!true))))= 1
((true^false) V (false V true))= 1
((!false)V(true^(!((false^(!true))V(!true)))))= 1
```

Вывод.

В результате лабораторной работы был получен опыт по работе с абстрактными типами данных, в частности со стеками. Были закреплены знания по написанию шаблона класса.

ИСХОДНЫЙ КОД

```
#INCLUDE <IOSTREAM>
#INCLUDE <SSTREAM>
#INCLUDE <FSTREAM>
#INCLUDE <CSTDLIB>
#INCLUDE <CCTYPE>
#INCLUDE <STRING>
#INCLUDE <IOMANIP>
#INCLUDE <ALGORITHM>



TEMPLATE <TYPENAME T>
CLASS VSTACK {

 PRIVATE:
        T* MEM;
        INT TOP;
        INT N;

 PUBLIC:
        VSTACK(INT LENGTH) {
                N = LENGTH;
                MEM = NEW T[N];
                TOP = 0;
        }

        INT EMPTY() CONST {
                RETURN TOP == 0;
        }

        VOID PUSH(T ELEM){
                MEM[TOP++] = ELEM;
                IF (TOP > N)
                        STD::COUT << "INCORRECT" << STD::ENDL;
        }
```

```
        VOID POP(){
                TOP--;
        }
        T GET_TOP() {
                IF (!EMPTY())
                        RETURN MEM[TOP-1];



        }
        VOID PRINT_STACK(){
                FOR (INT I = N - 1; N >= 0; N--)
        STD::COUT << "|" << STD::SETW(5) << MEM[N] << STD::ENDL;
        }
        ~VSTACK() {
                DELETE [] MEM;
        }

};

BOOL READOPERAND(STD::STRINGSTREAM &SS, CHAR INPUT){
        BOOL B;
        IF (INPUT == 'T') B = TRUE;
        ELSE IF (INPUT == 'F') B = FALSE;
        ELSE {STD::COUT<<"INCORRECT\N"; EXIT(0);}

        WHILE (INPUT!='E'){
                SS >> INPUT;
        }
        RETURN B;
}

BOOL EXECOPERATION(CHAR SIGN, BOOL OP1, BOOL OP2) {
        IF (SIGN == 'V')
                RETURN OP1 || OP2;
        ELSE IF(SIGN == '^')
                RETURN OP1 && OP2;

}
VOID READLOGICALEXPR(VSTACK<BOOL> &STACK, STD::STRINGSTREAM &SS) {
```

```
CHAR INPUT, SIGN;
INT FLAG = 0;
BOOL LEFTOP, RIGHTOP, INVERT = FALSE;

WHILE (SS >> INPUT) {

        IF (INPUT == '(') {
                READLOGICALEXPR(STACK, SS);
        }
        ELSE IF (INPUT == '!') {
                INVERT = TRUE;
                CONTINUE;
        }

        ELSE IF (INPUT == ')') {
                IF (!INVERT) {
                        RIGHTOP = STACK.GET_TOP();
                        STACK.POP();
                        LEFTOP = STACK.GET_TOP();
                        STACK.POP();
                        STACK.PUSH(EXECOPERATION(SIGN, LEFTOP, RIGHTOP));
                }
                ELSE {
                        BOOL A = STACK.GET_TOP();
                        STACK.POP();
                        STACK.PUSH(!A);
                }RETURN;
        }
        ELSE IF (INPUT == 'V' || INPUT == '^')          SIGN = INPUT;



        ELSE {
                IF (!FLAG) {
                        LEFTOP = READOPERAND(SS, INPUT);
                        STACK.PUSH(LEFTOP);
                        FLAG = 1;
                        CONTINUE;
```

```
                        }
                        ELSE {
                                RIGHTOP = READOPERAND(SS, INPUT);
                                STACK.PUSH(RIGHTOP);
                        }
                }
        }

}

VOID READFROMFILE(STD::IFSTREAM &FILE){



        STD::STRINGSTREAM SS;
        IF(!FILE.IS_OPEN()){
                        STD::COUT<<"НЕВЕРНЫЙ ПУТЬ К ФАЙЛУ\N";
                        RETURN;
                }

                STD::STRING STR;
                WHILE (STD::GETLINE(FILE,STR)){
                        VSTACK<BOOL> STACK(10);
                        SS << STR;
                        READLOGICALEXPR(STACK, SS);
                        STD::COUT << STR << "= " << STACK.GET_TOP() << STD::ENDL;
                        SS.CLEAR();
                        //STACK.PRINT_STACK();


                }

}
INT MAIN(INT ARGC, CHAR* ARGV[]) {
```

```
    IF (ARGC == 2) {

        STD::IFSTREAM FILE(ARGV[1]);
        READFROMFILE(FILE);

    }
    ELSE {
        VSTACK<BOOL> STACK(10);
        STD::STRING S;
        STD::STRINGSTREAM SS;
        STD::COUT << "ВВЕДИТЕ ЛОГИЧЕСКОЕ ВЫРАЖЕНИЕ:";
        STD::GETLINE(STD::CIN, S);
        S.ERASE(STD::REMOVE_IF(S.BEGIN(), S.END(), ISSPACE),S.END());
        SS << S;
        READLOGICALEXPR(STACK, SS);
        STD::COUT << "= " << STACK.GET_TOP() << STD::ENDL;
        //STACK.PRINT_STACK();



    }
    RETURN 0;
}
```